OULUN SEUDUN
AMMATTIKORKEAKOULU

Tuukka Saukkonen

TYÖNJOHTAJAN KALENTERIJÄRJESTELMÄN VAATIMUS-MÄÄRITTELY

TYÖNJOHTAJAN KALENTERIJÄRJESTELMÄN VAATIMUS-MÄÄRITTELY

Tuukka Saukkonen
Opinnäytetyö
Kevät 2012
Tietotekniikan koulutusohjelma
Oulun seudun ammattikorkeakoulu

TIIVISTELMÄ

Oulun seudun ammattikorkeakoulu
Tietotekniikka, ohjelmistojenkehitys

Tekijä(t): Tuukka Saukkonen
Opinnäytetyön nimi: Työnjohtajan kalenterijärjestelmän vaatimusmäärittely
Työn ohjaaja(t): Pekka Alaluukas
Työn valmistumislukukausi ja -vuosi: Kevät 2012 Sivumäärä: 33

Sähkö-Nero on oululainen kasvava sähköalan yritys, joka tuottaa asiakkailleen sähköalan suunnittelu- ja asennuspalveluita. Asiakaskunnan lisäännyttyä työnjohtajan työ on muuttunut liian haastavaksi hallita nykyisellä menetelmällä.

Tämän opinnäytetyön tarkoitus on tehdä vaatimusmäärittely työnjohtajan kalenterijärjestelmästä, jonka tavoitteena on osaltaan tehostaa työnjohtajan työtä. Kalenterijärjestelmä on kalenteri, jossa näkyy kaikkien asentajien työtehtävät viikkotasolla, ja samalla järjestelmä pitää yllä asiakas- ja työkohderekisteriä. Järjestelmästä ei ole tarkoitus tehdä teknistä vaatimusmäärittelyä, vaan määrittelyt, joiden pohjalta voidaan pyytää tarjouksia joko valmisjärjestelmistä tai yritykselle räätälöitävistä järjestelmistä.

Vaatimusmäärittelyssä käytettiin Soren Lauesenin esittämää kaksivaiheista vaatimusmäärittelyprosessia. Vaatimusten tueksi toteutettiin käyttöliittymäkuvat järjestelmän hahmottamiseksi ja käytettävyystestejä varten. Paperiprototypoimalla saatiin testattua järjestelmän toimivuus. Järjestelmä toimii erinomaisesti käytettävyystesteissä.

Kalenterijärjestelmä on ensimmäinen vaihe kohti lopullista toiminnanohjausjärjestelmää. Järjestelmään on mahdollista lähteä kehittämään muita liityntöjä, kuten laskutus tai varastonhallinta.

Asiasanat: Vaatimusmäärittely, käyttöliittymä, käytettävyystestaus

ABSTRACT

Oulu University of Applied Sciences
Information Technology, Software Development

Author(s): Tuukka Saukkonen
Title of thesis: Requirement specification of foreman's calendar system
Supervisor(s): Pekka Alaluukas
Term and year when the thesis was submitted: Spring 2012 Pages: 33

This thesis have been done for Sähkö-Nero. They are a rapidly growing electrical installation company and now they need a new system to plan their work more efficiently.

In this thesis we have done the requirement specification for the new system of foreman's calendar. The goal of the calendar system is to support the work of foreman. This requirement specification have been done by using the two-step approach present by Soren Lauesen. The outcome is not the technical requirements meant for software developers as such but specification for competitive bidding of commercial of the self or tailor-made systems. Paper prototyping was used to define the user interface of the system. Several paper prototyping cycles were implemented together with usability tests with end users to find the final specification of the user interface.

Calendar system is the first part of the targeted enterprise resource planning (ERP) system. The next steps will be developing interfaces to billing and material inventory systems.

Keywords: Software requirements, interface, usability test

SISÄLLYS

1 JOHDANTO

Kalenteri on tärkeä työkalu työnjohtajan päivittäisessä toiminnassa. Nykyaikaiset asiakasrekisterit alkavat olla digitaalisessa muodossa lähes jokaisessa pk-yrityksessä. Kalenterin tietoja voidaan päivittää suoraan kalenterirekisterien tietokannoista ja työnjohtajalle kalenterijärjestelmä on päivittäinen työkalu. Nykyään tarjotaan erilaisia valmiita järjestelmiä. Ne ovat useimmiten liian laajoja ja hankalia ottaa käyttöön pienessä yrityksessä, jolle tietotekniset asiat ovat jo muutenkin hankalia ymmärtää.

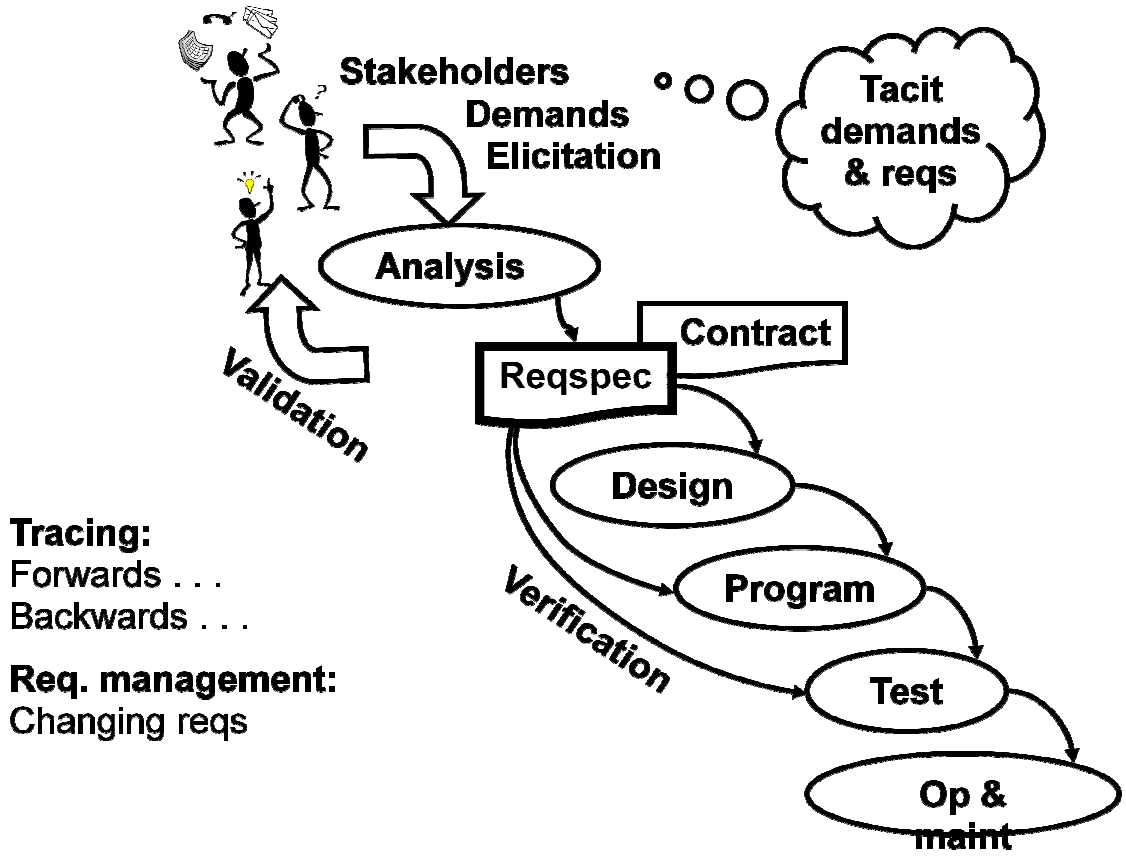
Työn tavoitteena on määritellä Sähkö-Nerolle uusi kalenterijärjestelmä, joka helpottaa työnjohtajan työn tekemistä. Suunniteltavan järjestelmän tavoitteena on saada helppokäyttöinen, toimiva ja työnjohtajan työtä tukeva järjestelmä. Web-pohjaisen käyttöliittymän avulla järjestelmä toimii jokaisella käytetyimmällä selaimella ja se ei ole käyttöjärjestelmästä riippuvainen. Suunniteltava kalenterijärjestelmä muodostaa kattavan ominaisuuden työnjohtajan työn tukemiseen. Se auttaa työnjohtajaa seuraamaan ja päivittämään asentajien työkohteita ja työkohteiden vaiheita. Lisäksi siinä on valmiina asiakas- ja työkohderekisterit.

Vaatimusmäärittely on tärkeä osa järjestelmien toteuttamisessa. Vaatimusmäärittelyssä työn tuloksena syntyy vaatimusmäärittelydokumentti, jonka pohjalta tilaaja saa yksityiskohtaisen selvityksen kalenterijärjestelmän tarpeista. Sen sisältö koostuu järjestelmän toiminnallisista ja ei-toiminnallisista vaatimuksista sekä käyttöliittymästä. (Ohjelmiston vaatimusmäärittely. 2012.) Tässä työssä ei ole keskitytty tekniseen määrittelyyn, vaan määrittelyihin, joiden pohjalta voidaan pyytää tarjouksia joko valmisjärjestelmistä tai yritykselle räätälöitävistä järjestelmistä. Tilaajan omalle vastuulle jää tarjouksien hakeminen valmiista tuotteista tämän vaatimusmäärittelyn pohjalta tai kehityksen jatkaminen omin avuin. Alkuperäinen vaatimusmäärittelydokumentti on vain tilaajan käytössä.

2 VAATIMUSMÄÄRITTELY

2.1 Vaatimusmäärittelyn aktiviteetit

Vaatimusmäärittely koostuu kuudesta eri aktiviteetista. Näitä kutsutaan englannin kielessä nimeltä Elicitation, Documentation, Negotiation, Validation ja Verification sekä Management. Kuvan 1 analyysi sisältää Negotiation-vaiheen.



KUVA 1. Vaatimusmäärittelyn toteutus (Lauesen 2002, 5)

Documentation tarkoittaa analyysivaiheen lopputuloksista syntyvän vaatimusmäärittelyn tuottamista. Sana Elicitation tarkoittaa aktiviteetteja, joiden avulla hankitaan ymmärrys rajatun järjestelmän käyttäjän tarpeista. Tämä johtaa siihen, että määrittely paljastaa vaatimukset ja rajaukset eri lähteistä, kuten sidosryhmistä, käyttäjien asiakirjoista, lainsäädännöstä ja standardeista. Tällä myös lisätään uudenlaista ja innovatiivista kehitystä aiottuun tuotteeseen. (Carroll 1995; Weinberg 1988.)

Documentation eli vaatimusten spesifikaatio on hyvin jäsennelty asiakirja, josta löytyy kaikki mahdollinen informaatio vaatimuksista. Lopullinen vaatimusmäärittelydokumentti luo perustan myöhemmälle kehitysvaiheelle tai tuotteen vaihdolle. Erilaisten sidosryhmien välillä, kuten asiakkaitten ja suunnittelijoiden välillä, on mahdollista käyttää vaatimusten dokumentoinnissa erilaisia esitysmuotoja. (Pohl 1994.)

Negotiationin päämäärä on toteuttaa riittävän yksimielinen, eri sidosryhmien keskuudessa hyväksyttävissä oleva vaatimusmäärittely. Vaatimusmäärittely on vakaampi kehitysvaiheen aikana, jos riittävä hyväksyntä saavutetaan. Ilman sopimusta vaatimuksista hanke todennäköisesti epäonnistuu, kun esimerkiksi aika loppuu kesken. (Wiegers 1999.)

Validationin ja verificationin tavoitteena on todentaa, että järjestelmävaatimukset ovat selkeitä, täydellisiä, oikeita ja ymmärrettäviä. Validation varmistaa, että oikeat vaatimukset on dokumentoitu vaatimusmäärittelyssä, jossa on kuvattu sellainen järjestelmä, jonka sidosryhmät haluavat. Verification varmistaa, että dokumentoitu vaatimusmäärittely on tehty oikein ja kaikki kuvaukset ovat johdonmukaisia, ristiriidattomia ja yksikäsitteisiä. (Thayer – Dorfman 1997; Sommerville – Sawyer 1997.)

Managementin päämäärä on huolehtia siitä, että vaatimuksia käydään läpi koko kehityksen ja järjestelmän elinkaaren aikana ja täten varmistaa, että johdonmukaiset ja ajantasaiset vaatimusmäärittelyt on saatavilla koko ajan. Tätä tuetaan ylläpitämällä jäljitystietoa vaatimusten lähteiden ja niiden toteutuksen välillä. Vaatimusten hallinnan vastuulla on valvoa, että tarvittava jäljitystieto on johdonmukaisesti ajan tasalla. Vaatimusmäärittelyn ajantasaisuus on edellytys myöhemmille julkaisuille, vian korjauksille ja vaatimusten uudelleenkäytölle. (Thayer and Dorfman 1997; Hull et al. 2002.)

2.2 Vaatimusmäärittelyn lähtökohdat

Vaatimusmäärittelyä voidaan lähteä tekemään kahdella eri tavalla. Jos vaatimuksia käytetään tarjouspyynnön pohjana valittaessa joukosta valmisjärjestelmiä, ei ole syytä esittää vaatimuksia liian yksityiskohtaisina ja teknisellä tasolla, koska silloin voidaan rajata tarjouksista ulos monia käyttökelpoisia ratkaisuja.

Vaatimuksien toimiessa suoraan ohjelmiston kehittäjien suunnittelun lähtökohtana tarvitaan hyvin yksityiskohtaiset vaatimukset. (Lauesen 2002, 35.)

COTS on lyhenne englannin kielen sanoista "Commercial Off The Self" ja se tarkoittaa kaupallista tuotetta jonka voi ostaa, esimerkiksi Microsoft Office, Windows 7 tai Norton Antivirus. Määriteltäessä järjestelmää, jonka pohjalta haetaan tarjouksia valmiista järjestelmistä, voidaan puhua COTS-tyyppisestä määrittelytavasta. Kun ostetaan tiettyä tuotetta, on hyvä olla perillä omista tarpeista, jotta osataan valita vaihtoehdoista paras. (Lauesen 2002, 9.)

Kun asiakas hakee valmista ohjelmaa vaatimusmäärittelyn pohjalta, sen ei tarvitse olla määriteltynä kovin tarkkaan. Vaatimusmäärittelyksi riittää, kun asiakas tietää käyttäjien tehtävät, joita ohjelmalla on tarkoitus tukea. Tällaisia vaatimuksia kutsutaan domain-tason vaatimuksiksi. (Lauesen 2002, 9.) Esimerkkejä domain-tason vaatimuksista ovat seuraavat:

- ohjelman tulee tukea asiakastietojen tallentamista
- ohjelman tulee tukea työkohteiden määrittelyä
- ohjelman tulee tukea työvaiheiden jakamista asentajille.

Näillä vaatimuksilla voidaan pyytää tarjouksia, joita voidaan arvioida suhteessa esitettyihin vaatimuksiin.

Kun vaatimukset menevät suoraan ohjelmistojen kehittäjien tarpeeseen, tulee vaatimusten olla hyvin yksityiskohtaisia. Yksityiskohtaisella tarkoitetaan tuotteen ominaisuuksia. Esimerkkejä edellä mainituista vaatimusmäärittelyä vastaavan tuotteen ominaisuuksista olisivat (ns. product-tason vaatimuksia) seuraavat:

- ohjelmassa tulee olla toiminto, joilla asiakkaan tiedot tallennetaan
- ohjelmassa tulee olla toiminto, joilla määritetään työvaiheita työkohteelle
- ohjelmassa tulee olla toiminto, joilla voidaan jakaa tehtäviä asentajille.

2.3 Vaatimusmäärittelyn abstraktiotasot

Vaatimuksia voi esitellä useammalla eri abstraktiotasolla: goal, domain, product ja design. Näiden vaatimustasojen avulla järjestelmä saadaan rajattua haluttuun konseptiin.

2.3.1 Goal-taso

Goal-tasolla kuvataan ne asiakkaan liiketoiminnalliset tavoitteet, joiden saavuttamista hankittavan järjestelmän tulee tukea ja edesauttaa (Lauesen 2002, 25). Tällä tarkoitetaan esimerkiksi sitä, miten yritys hyötyy ohjelmasta, jotta sen liiketoiminnallinen kehitys menisi eteenpäin tulevaisuudessa.

2.3.2 Domain-taso

Domain-tason tarkoitus on selvittää ja kuvata, mitä käyttäjän tehtäviä uudella järjestelmällä on tarkoitus tukea. Domain koostuu tuotteesta, käyttäjistä ja näiden ympäristöstä. (Lauesen 2002, 21.) Esimerkkinä tulevassa kalenterisovelluksessa käyttäjän pitää nähdä viikkotasolla asentajille merkityt työkohteet ja pystyä muuttamaan niitä haluamansa mukaisesti.

2.3.3 Product-taso

Product-tasolla määritellään, mitä toimintoja tuotteessa on sekä mitä syötteitä ja tulosteita näihin toimintoihin liittyy (Lauesen 2002, 26). Esimerkiksi tuotteessa on oltava toiminto, jossa hakuavaimen perusteella voidaan hakea tietyn asiakkaan kaikki työkohteet.

2.3.4 Design-taso

Design-tasolla kuvataan tuotteen toiminnallisuuksien ja laatuvaatimuksien toteuttamista yksityiskohtaisella tasolla (Lauesen 2002, 26). Edellisen esimerkin määrittely design-tasolla voisi olla seuraava: Käyttöliittymästä löytyy ikkuna, jossa on mahdollista tehdä hakuja järjestelmään. Haun yhteydessä on mahdollisuus valita, haetaanko tietoa asiakkaista vai työkohteista. Tässä ei kuitenkaan kerrota, miten tuotteen sisällä toimiva kontrolleri toteutetaan.

2.4 Vaatimusmäärittelyn lähestymistavat

Vaatimusmäärittelyn tyypillisimmät lähestymistavat Lauesenin esittämistavan mukaan ovat traditionaalinen, nopea ja kahden askeleen lähestymistapa.

2.4.1 Traditionaalinen lähestymistapa

Traditionaalinen lähestymistapa keskittyy tuotetason vaatimuksiin. Traditionaalinen lähetystapa on hyvin yleinen malli tuotekehitystä varten tehtävässä tuotteen määrittelyssä. Määriteltäessä tuotetta tehdään haastatteluja sidosryhmille, tutkitaan saatuja dokumentteja, järjestetään työryhmäistuntoja ja aivoriihiä. Määrittelyn tuloksena syntyy seuraavia osia: johdanto-osat, järjestelmän rajaus, tietovaatimukset, tuotetason vaatimukset ja laatuvaatimukset. (Lauesen 2002, 31.)

Johdanto-osa sisältää liiketoiminnalliset tavoitteet. Rajattaessa järjestelmää otetaan huomioon, mitä järjestelmästä jätetään pois ja mihin järjestelmä rajataan. Tietovaatimukset kuvataan tietomallin ja data dictionaryn avulla. Tuotetason vaatimukset saadaan käyttäjätehtävälistauksista, yksityiskohtaisista vaatimuksista ja tekstuaalisesta järjestelmäkuvauksesta. Laatuvaatimuksilla kuvataan kriittisiä laatutekijöitä. (Lauesen 2002, 31.)

2.4.2 Nopea lähestymistapa

Nopea lähestymistapa tuottaa domain-tason vaatimukset. Nopeassa lähestymistavassa on tarkoitus tehdä yhteistyötä asiantuntijan kanssa ja löytää käyttäjätehtävien kuvaukset vaatimusmäärittelydokumenttiin. Vaatimusmäärittelydokumentin ensimmäisessä versiossa ei ole tarvetta tietää liiketoiminnallista tavoitetta, koska päätös uudesta tuotteesta on jo tehty. Vaatimusmäärittelyn lopullista versiota varten tehdään haastatteluja ja pidetään aivoriihiä, joista muodostuu liiketoiminnallinen tavoite. (Lauesen 2002, 34.)

Johdanto-osa sisältää liiketoiminnalliset tavoitteet ja BPR (Business Process Re-engineering) -tehtävät eli järjestelmän käyttöönoton seurauksena uudistuneen työprosessin mukaiset tehtävät vaatimusmäärittelydokumentin lopullisessa versiossa. Tehtävien kuvausten ja niitä tukevien vaatimusten tulee olla selvitettynä. Liiketoiminnallisia tavoitteita verrataan annettuihin tehtäviin lopullisessa versiossa. Järjestelmä rajataan kontekstikaavion mukaiseen rajapintaan ja toimintoihin. Tietovaatimusten ja kriittisten laatutekijöiden laatuvaatimusten tulee olla selvitettynä. (Lauesen 2002, 34.)

2.4.3 Kahden askeleen lähestymistapa

Kahden askeleen lähestymistapa sisältää domain- ja design-tason vaatimukset. Ensimmäisenä lähestytään domain-tason vaatimuksia. Ensin selvitetään ja kuvataan täsmällinen liiketoiminnallinen tavoite, tietovaatimukset, tehtäväkuvaukset jne. Tämän jälkeen suoritetaan arvioita ja tarkistuksia. (Lauesen 2002, 35–36.)

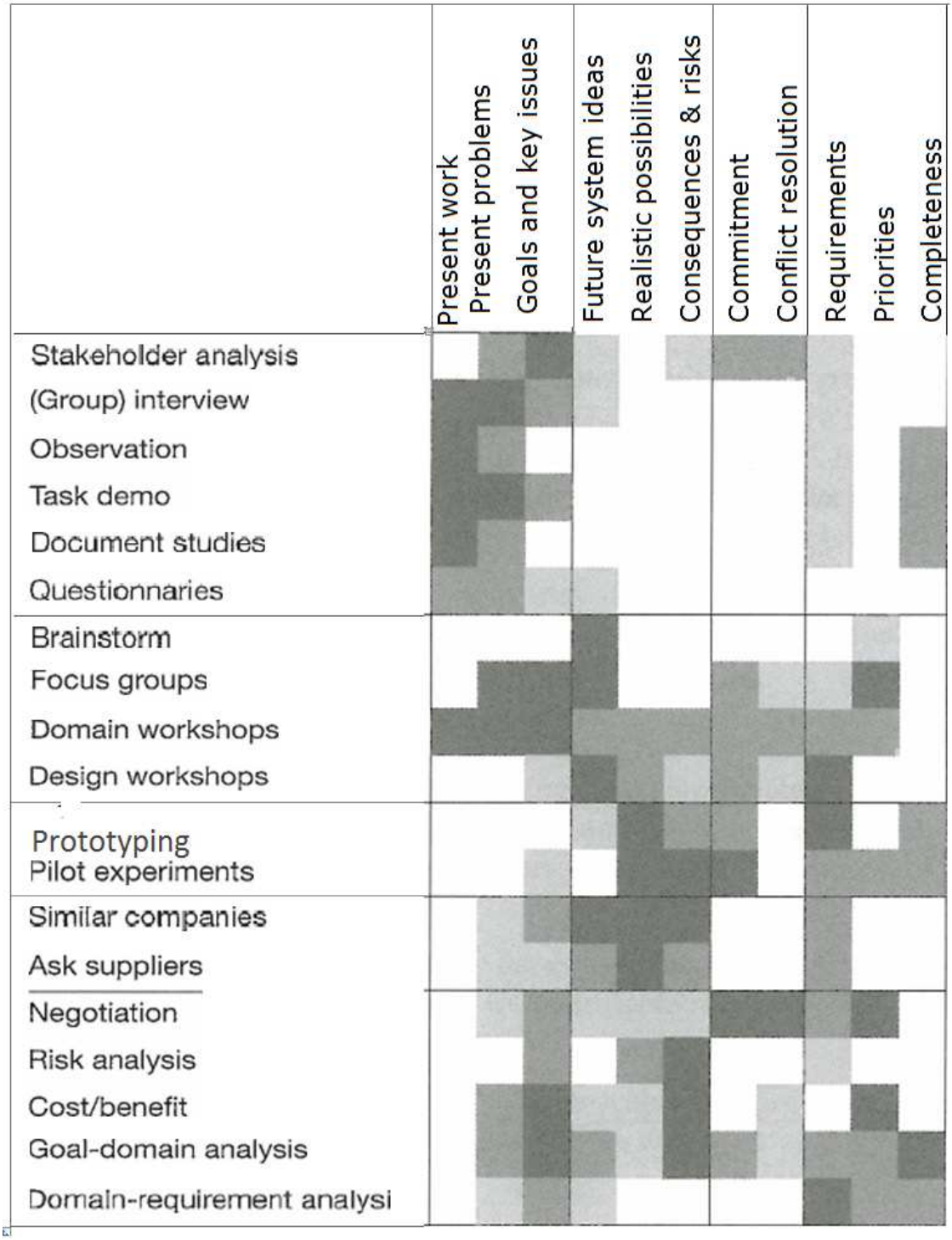
Seuraavalla askeleella selvitetään design-tason vaatimukset, jotka tuotetaan domain-tason vaatimuksista monimutkaisille liitynnöille. Esimerkiksi käyttöliittymä voi usein olla tällainen monimutkainen liityntä käyttäjän ja järjestelmän välillä. Käyttöliittymäkuvaukset saadaan aikaan hyödyntämällä tehtävien ja tietojen kuvauksia. Käytettävyystestit paljastavat, onko käyttöliittymää tarpeeksi helppo oppia käyttämään ja tukeeko se annettujen tehtävien suorittamista. Käytettävyystestien tarkoitus on tarkistaa ja vahvistaa vaatimukset. Kehittäjät suunnittelevat ja määrittelevät monimutkaisille teknisille liittymille yhteisiä formaatteja, protokollia jne. On myös hyvä tehdä toiminnallista prototyyppitestausta, josta saadaan selville, toimivatko kaikki tehtävät hyvin yhdessä. (Lauesen 2002, 35–36.)

Kahden askeleen lähestymistapa soveltuu monentyyppisiin projekteihin: talon sisäiseen käyttöön tuleviin järjestelmiin, jatkuvaan kehitykseen liittyviin projekteihin ja räätälöityihin järjestelmiin. Kahden askeleen malli ei sovi COTS-pohjaisiin systeemeihin, koska niillä on omat käyttöliittymät ja uudelleen määritellyllä käyttöliittymällä ei ole merkitystä tai se voi olla suorastaan haitallinen rajatessaan monia vaihtoehtoisia COTS-pohjaisia järjestelmiä ulos vertailusta. (Lauesen 2002, 35–36.)

2.5 Vaatimusmäärittelyn tekniikat

Vaatimusmäärittely voidaan toteuttaa käyttämällä eri tekniikoita tai yhdistämällä näitä kaikkia tarpeen mukaan. Vaatimuksia ei voida määrittää koskaan täydellisesti, koska aina on olemassa ns. hiljaisia vaatimuksia. Hyvä asiantuntija tietää useita tekniikoita, mutta myös tietää, missä käyttää niitä ja missä jättää käyttämättä. Asiantuntija osaa muuttaa ja yhdistellä tekniikoita tarpeen vaatiessa. Vaatimusmäärittelyyn löytyy tekniikoita seuraaviin osa-alueisiin: elisitointi, vali-

dointi, verifiointi ja jäljitys. Kuvassa 2 esitetään taulukko tekniikoista, joilla vaatimuksia voidaan selvittää. Mitä tummempi ruutu, sitä paremmin se antaa tietoa tuotteen kehittämiseen. (Lauesen 2002, 1–2.)



KUVA 2. Vaatimusmäärittelyn eri tekniikat (Lauesen 2002, 338)

Elisitoinnin tarkoitus on etsiä ja rakentaa vaatimuksia. Vaatimukset saadaan käyttäjiltä ja sidosryhmiltä. Sidosryhmillä on aina tarpeita, joita asiantuntijat elisitoivat ja muodostavat näistä vaatimuksia. Sidosryhmät muodostuvat erilaisista käyttäjien rooleista, kuten tilaajasta, tilaajan IT-osastosta jne. Sidosryhmään kuuluu ihmisiä, jotka ovat osallisena tuotteeseen. Sidosryhmä koostuu kahdesta eri pääryhmästä, asiakkaista ja kehittäjistä. Elisitointi on hankala prosessi, koska sidosryhmien voi olla hankala ilmaista omia tarpeitaan ja vaatimuksiaan. Käyttäjillä voi olla ongelma ottaa käyttöön uusia toimintoja ja tapoja. IT-puolen käyttökokemus voi olla olematonta ja uusi tuote voi jäädä käyttämättä. Sidosryhmällä tarpeet muuttuvat kokoajan, kun he voivat kuvitella tarvitsevansa jotain uutta ja hienoa, jota he ovat nähneet jollain toisella olevan käytössä. (Lauesen 2002, 4.)

Osa käyttäjien tarpeista voi olla hankala kirjoittaa ylös ilman, että suunnittelisi ratkaisua kokoajan. Tästä tuloksena on, että todelliset tarpeet eivät vastaa kirjoitettuja vaatimuksia. Tämän vuoksi on tärkeää, että sidosryhmät tarkistavat, että vaatimukset vastaavat todellisia tarpeita. Tällaista menetelmää kutsutaan validoinniksi. (Lauesen 2002, 4.)

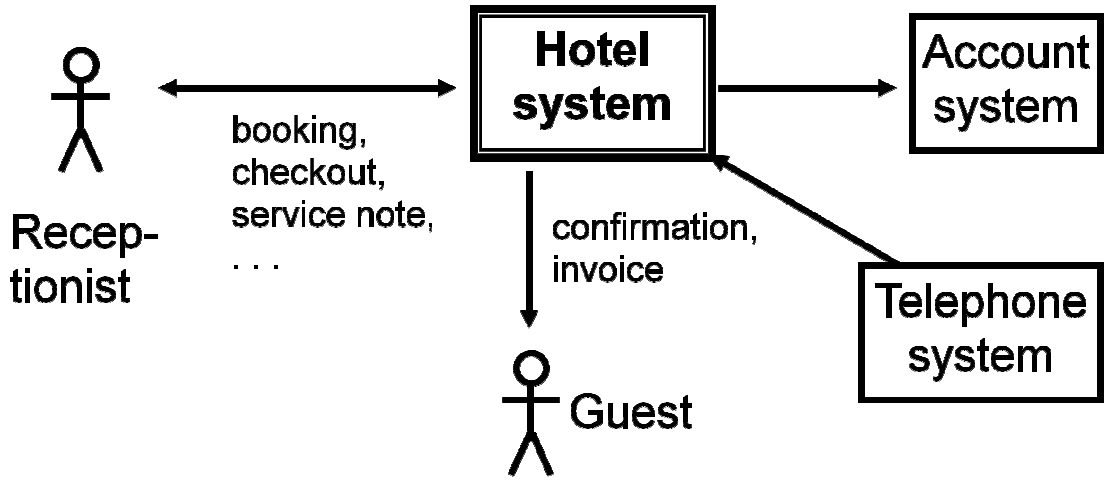
Kun tuotetta verifioidaan, sillä tarkistetaan, että tuote täyttää sille määritetyt vaatimukset. Tämä tapahtuu vähintään hyväksyttävästi silloin, kun tuote testataan hyväksyttävyystesteissä. Näissä testeissä jokainen vaatimus käydään läpi yksitellen ja tarkistetaan, että tuote täyttää vaatimukset. (Lauesen 2002, 5–6.)

Vaatimusten jäljitystä tarvitaan, kun verrataan vaatimuksia muuhun informaatioon. Kun seurataan vaatimusten toteutusta ohjelmassa, tarkistetaan, että vaatimukset on otettu huomioon ja kaikki tarpeet on huomioitu vaatimuksissa. Kun ohjelma on valmis, jäljitetään vaatimuksia, että kaikilla tarpeilla on päämäärä, eli ne liittyvät jonkun liiketoiminnallisen tavoitteen (goal) saavuttamiseen. Jäljitettäessä vaatimuksia ohjelmasta tarkistetaan, että kaikki ohjelman osat ovat vaadittuja. (Lauesen 2002, 6.)

2.6 Vaatimusmäärittelyn kuvaustekniikat

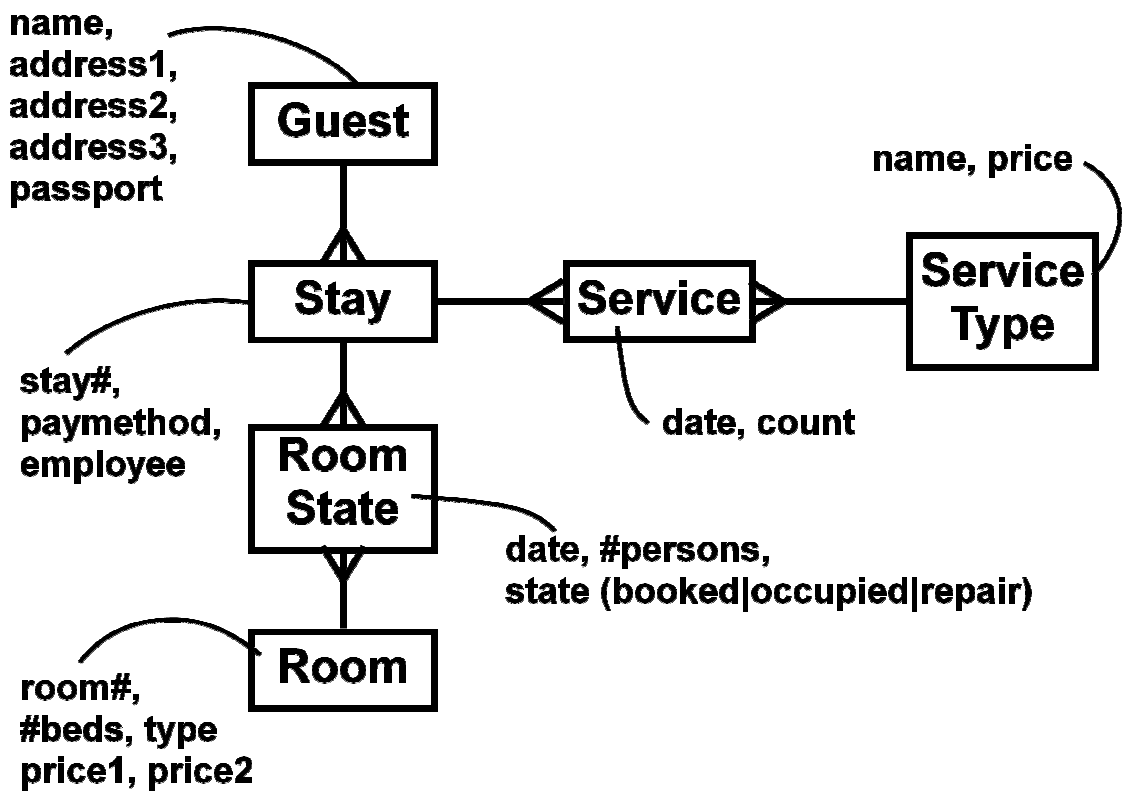
Vaatimusmäärittelyssä käytetään useampaa eri tekniikkaa kuvaamaan tuotetta. Esittelen seuraavaksi muutaman tärkeän ja hyödyllisen kuvaustavan Lausenin mukaan.

Kontekstikaavio kuvaa tuotetta ja sen ympäristöä. Ympäristö koostuu eri tekijöistä, kuten ihmisistä ja muista järjestelmistä. Kontekstikaavio kuvaa erinomaisesti tuotteen rajapinnan näkymää. Se kuvaa hyvin, mitä uusi tuote sisältää ja mikä on ulkopuolista. Projektin alkupäässä on erittäin hyödyllistä hahmotella kontekstikaavio ja päivittää sitä tutkimuksen aikana. Kontekstikaavion tekeminen voi paljastaa, miten isosta projektista on kyseessä. Kuva 3 esittää esimerkin kontekstikaaviosta. Siinä Hotel system on kehitettävä järjestelmä ja se liittyy kirjanpito- ja puhelinjärjestelmään, mutta näiden järjestelmien kehittäminen ei kuulu projektin piiriin – ainoastaan niihin tarvittavat liittymät on toteutettava. (Lauesen 2002, 76–77.)



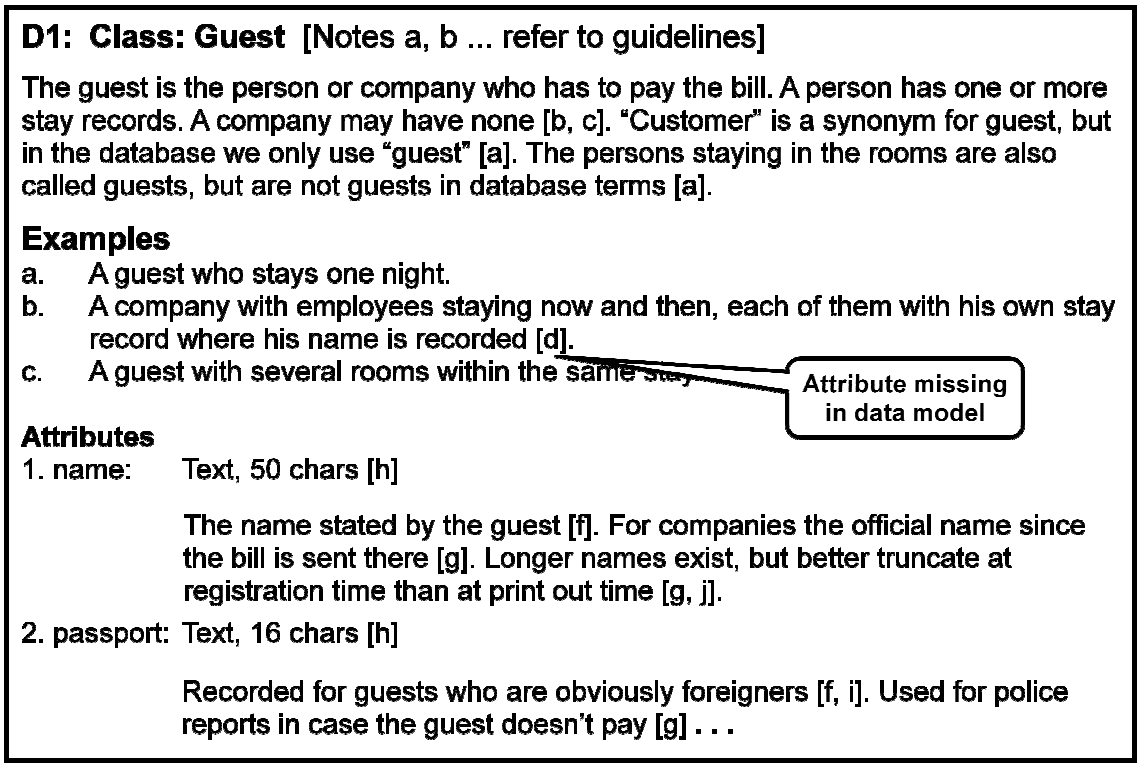
KUVA 3. Kontekstikaavio (Lauesen 2002, 77)

Luokkakaavio on datamalli, joka koostuu useasta tiedosta, jonka avulla voidaan tehdä jotakin. Jokainen luokka ei ainoastaan varastoi tietoa, vaan se sisältää myös toimintoja, jotka toimivat näiden tietojen mukaan. Luokkakaaviot ovat laajassa käytössä. Asiantuntijan pitäisi tuntea luokkakaaviot, vaikka ne ovat enemmänkin ohjelmaluonnoksia kuin vaatimuksia. Kuvassa 4 esitetään esimerkki luokkakaaviosta. (Lauesen 2002, 45, 182.)



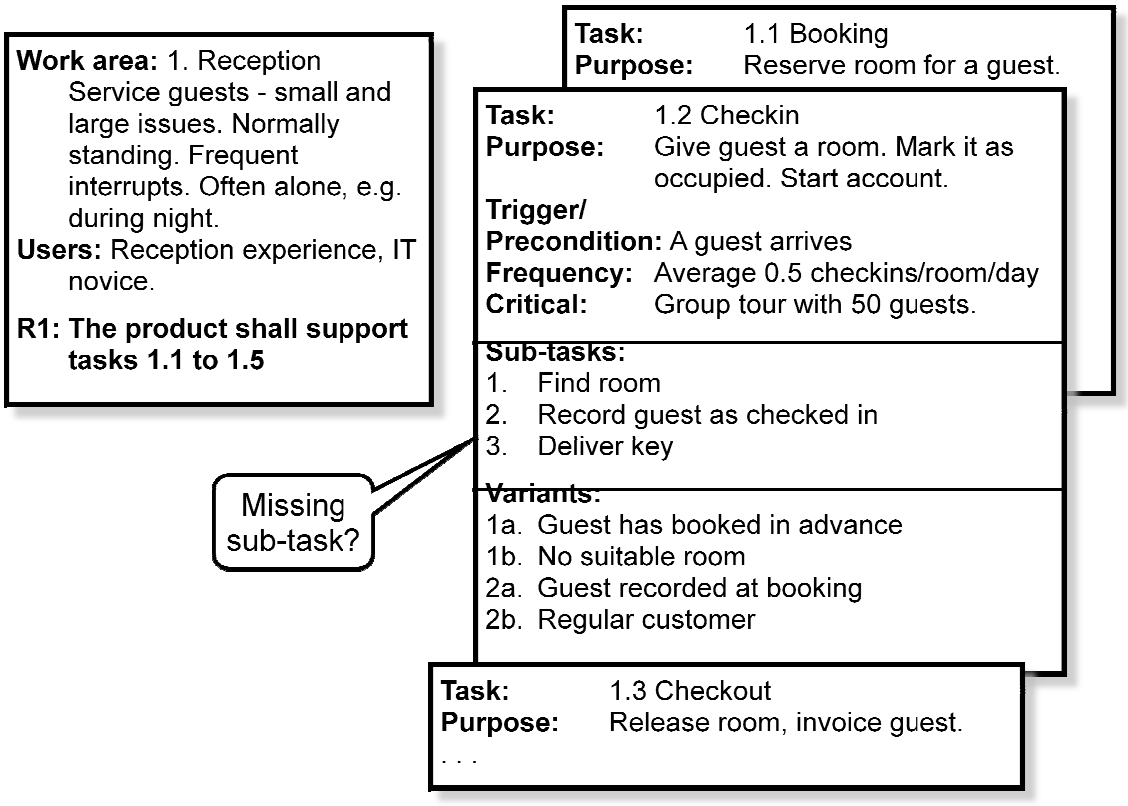
KUVA 4. Luokkakaavio (Lauesen 2002, 45)

Data dictionary on tekstuaalinen kuvaus tuotteen sisäisistä ja ulkoisista tiedoista. Pelkistetyllä tekstuaalisella kuvailulla tiedot voidaan tallentaa järjestelmään. Kehittäjän työ on siirtää nämä tiedot datamalliin ja tietokantaan. Data dictionary on sanallista tietoa, jolla kuvataan rakennetta kokonaisuudessaan. Jos on olemassa tietomalli tai luonnos, kannattaa rakentaa data dictionary luokkamallin mukaan. Data dictionaryn avulla voidaan määrittää kaikki mahdolliset yksityiskohdat ja erityistapaukset. Se toimii parhaiten yhdistettynä datamallin kanssa. Suurin ongelma data dictionaryssa on se, että kirjoittamisessa menee paljon aikaa. Yksityiskohtaista tietoa voi löytää loputtomasti, jos vain tarpeeksi etsii, ja voi lopulta olla vaikea päättää, milloin tiedot ovat tarpeeksi hyviä. Validoinnin kannalta pitkä kuvailu on myös hankalaa. Kuvassa 5 on esitetty esimerkki Data dictionarystä. (Lauesen 2002, 56–59.)



KUVA 5. Data dictionary (Lauesen 2002, 57)

Task description koostuu käyttäjien tehtävien tekstuaalisesta kuvailusta. Käyttäjien ja kehittäjien on helppo ymmärtää sitä ja siitä on helppo määrittää variaatioita ja monimuotoisuutta. Task descriptionissa tarkistukset ovat helppoja. Nämä ovat domain-tason vaatimuksia. Domain-tason aktiviteetit on esitetty jo aiemmin ja vaatimukset tukevat näitä aktiviteetteja. Tässä selvitetään, miten ihmisen ja tietokoneen pitäisi toimia yhdessä, ja se ei kerro mitään tuotteen ominaisuuksista. Task description muistuttaa hyvin paljon käyttötapauskuvailua. Ero on siinä, kuvaillaanko, mitä ihminen tekee vai mitä tietokone tekee. Tehtävällä tarkoitetaan sitä, mitä käyttäjä ja tietokone tekevät yhdessä saavuttaakseen jonkun päämäärän. Käyttötapauksella tarkoitetaan lähinnä tehtävän muodostavia tuotteen osia. Ihmisen tehtävällä tarkoitetaan käyttäjien osaa tehtävissä. Kuvassa 6 esitetään esimerkki tehtäväkuvauksesta. (Lausen 2002, 92–99.)



KUVA 6. Task description (Lauesen 2002, 93)

Laatuvaatimuksien tarkoitus on määrittää, kuinka hyvin järjestelmä suorittaa sille asetettuja tehtäviä. Laatuvaatimuksia kutsutaan ei-toiminnallisiksi vaatimuksiksi verrattaessa toiminnallisiin vaatimuksiin. Laatuvaatimukset eivät ole ainoastaan ohjelmallisia vaatimuksia, vaan ne sisältävät myös laitteiston vaatimukset. Esimerkiksi turvallisuus riippuu yhtä lailla laitteistosta kuin ohjelmasta. Lauesen (2002) jakaa laatutekijät McCallin ja Matsumoton mukaan toimintaan, revisioon ja muutokseen.

Ennen kuin laatuvaatimuksia aletaan selvittää, on syytä analysoida niiden kunkin merkityksellisyyttä kehitettävässä järjestelmässä. Tämä voidaan tehdä taulukon 1 mukaisella matriisilla. Taulukon 1 esimerkkitapauksessa useimmat laatutekijät on todettu tavanomaisiksi. Tärkeitä tai erittäin tärkeitä ovat luotettavuus, käytettävyys ja yhteentoimivuus. Numeroilla viitataan tässä kommenttiin, joilla selvennetään, missä mielessä laatutekijät ovat tässä sovelluksessa tärkeitä. Esimerkiksi käytettävyyden osalta voitaisiin tarkentaa, että on erittäin tärkeää, että käyttöliittymä suunnitellaan helposti opittavaksi ja että työtehtävien suorittaminen sujuu nopeasti.

Laatumatriisin avulla voidaan kohdentaa laatuvaatimusten selvittäminen tärkeisiin laatutekijöihin, eikä kuluteta aikaa vähemmän tärkeään (Lauesen 2002, 217–227).

TAULUKKO 1. Laatutekijät vaatimusmäärittelyssä

Laatutekijät kalenteritoiminnalle	Erittäin tärkeä	Tärkeä	Tavallinen	Välttävä	Turha
TOIMINTA					
Turvallisuus			x		
Tarkkuus			x		
Luotettavuus		1			
Käytettävyys	2				
Tehokkuus			x		
REVISIO					
Ylläpito			x		
Testattavuus			x		
Joustavuus			x		
MUUTOS					
Siirrettävyys					x
Yhteentoimivuus		3			
Uudelleenkäytettävyys					x
Asennettavuus			x		

3 KÄYTTÖLIITTYMIEN SUUNNITTELU JA TESTAUS

Käyttöliittymäsuunnittelun peruslähtökohtana ovat käyttäjien tarpeet. Käyttöliittymäsuunnittelussa tarkastellaan käyttäjän toimintaa. Käytännön suunnittelu on usein teknistä suunnittelua, laitteiden ja komponenttien valintaa. Tämä näkyy usein käyttöliittymäsuunnittelun ongelmana. Monissa pieniresurssisissa yrityksissä tuotekehitys on ongelmallista, kun projekteihin ei saada tarpeeksi osaamista eri aihealueiden asiantuntijoista. Tästä syystä käyttäjän ja tuotteen yhteistoiminta kärsii. (Vuori – Kivistö-Rahnasto – Toivonen 2012, 1–3.)

Käyttöliittymän tarkoitus on esittää, miten joku laite toimii. Tarkoitus ei ole esittää laitteen teknisiä ominaisuuksia. Käyttöliittymissä tärkeää on käytettävyys eli se, miten laite toimii todellisen käyttäjän toimesta todellisessa tilanteessa. Kaikkein oleellisinta käyttöliittymässä on, miten siinä näkyy käyttäjän malli prosessista, eikä kehittäjän malli käytännön toteutustavasta. Tämä edellyttää kehittyneiden suunnittelutapojen käyttöä. (Vuori ym. 2012.)

Käyttöliittymä on hyvä, jos sitä on helppo oppia käyttämään ja sen käyttötapa on itsestään selvä. Järjestelmän käyttö on turvallista, se on nopea konfiguroida ja tuotannon laatu on korkeaa. Käyttöliittymän laatu on hyvä, kun tiedetään miten se sopii käyttäjälle, käyttötehtävien kokonaisuuteen ja käyttöolosuhteisiin. (Vuori ym. 2012.)

Kun tuotteesta kehitetään uutta teknologiaa hyödyntävä uuden sukupolven ratkaisu, liittyy tähän useita ongelmia. Uuden sukupolvenvaihdos on ongelmallista, kun otetaan käyttöön uutta teknologiaa. Sukupolvenvaihdos voi tapahtua, kun järjestelmän tekniikka päivitetään uudemmalla tekniikalla. Tämän johdosta uudistettu järjestelmä voi olla tehokkaampi ja luotettavampi. Ongelmia voi olla siinä, että ennen tarvittiin useampaa vipua ja uudessa versiossa kaikki tapahtuu samasta vivusta. (Vuori ym. 2012.)

3.1 Käyttöliittymä suunnittelu

Hyvän käyttöliittymäsuunnittelun avainasemassa on hyvin tehty vaatimusmäärittely. Vaatimusmäärittelyn pohjalta käyttäjä tietää jo enemmän, kuin että "ohjel-

man pitää olla helppokäyttöinen". Tämä ei kelpaa käyttöliittymän lähtökohdaksi. (Vuori ym. 2012.)

Käyttöliittymäsuunnittelun keskeisiä lähtökohtia on teknisen järjestelmän ja käyttäjän työnjako. Suunnittelun jokaisessa vaiheessa pitää ottaa huomioon käyttöliittymää koskevat ominaisuudet. Kehityksen aikana käsitykset muuttuvat järjestelmän ympärillä. Vaatimuksia ja oletuksia pitää päivittää projektin aikana, kun käsitykset ja oletukset järjestelmästä elävät koko suunnitteluprosessin ajan. Suunnittelun lähtökohta on voinut olla alun perin hyvin virheellinen ja se on muovautunut projektin aikana oikeaan formaattiin. (Vuori ym. 2012.)

Käyttäjän näkökulmasta voidaan kehittää kuvauksia järjestelmästä uuden tiedon lisääntyessä. Erilaiset simulaatiot ja testitapaukset ovat hyviä havainnollistamaan, millaisia ongelmia käyttäjätehtävissä voi tulla vastaan ja mitä puutteita laitteessa on. Kun järjestelmässä alkaa olla jotain oikein, voidaan aloittaa käyttökokeet. (Vuori ym. 2012.)

Seuraavissa alaluvuissa käsitellään lyhyesti web-pohjaisen järjestelmän tyypillisiä käyttöliittymän suunnitteluperiaatteita ja sivurakennetta. Tällä kuvataan käyttöliittymien yhdenmukaisuutta ja toimintatapoja suunnittelutyössä.

3.1.1 Sivujen ylä- ja alatunnisteet

Sivujen asemointi tapahtuu www-sivuja luotaessa seuraavista osista: ylä- ja alatunnisteet, navigointialueet ja työskentelyalueet. Sivun pinta-alasta 80 prosenttia jaetaan työskentelyalueeseen ja 20 prosenttia tunnistetiedoille ja navigointiapuvälineille. Ylä- ja alatunnisteet ovat joka sivulla samanlaiset. Ylätunniste sisältää seuraavat järjestelmän kannalta tarvittavat tiedot:

1. sovelluksen tunnus ja nimi
2. kielivalinnat
3. tekstiversio
4. palaute
5. ohjeet
6. hakutoiminnot
7. kirjautumistiedot

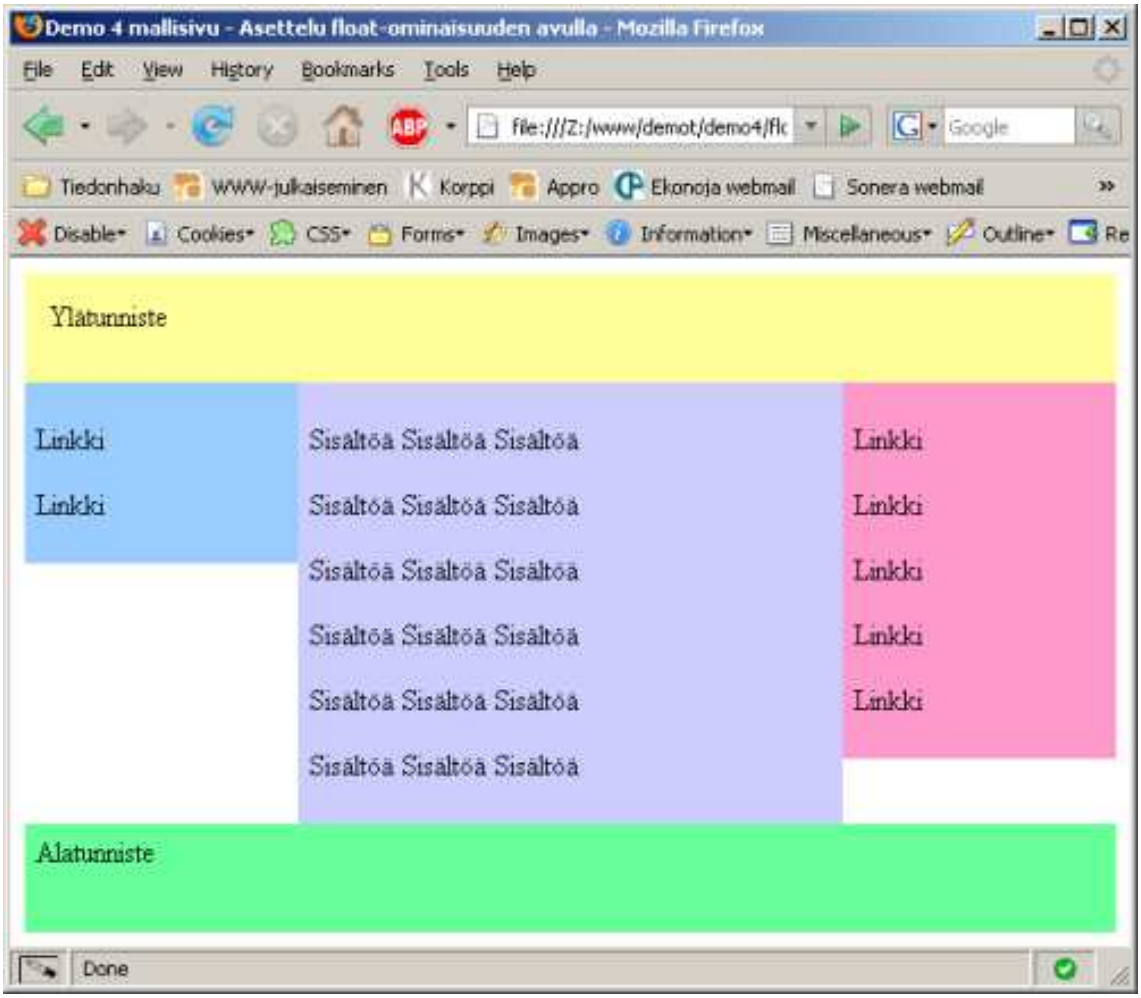
8. organisaation nimi
9. yhteystiedot.

Alatunniste sisältää tietoa, jota tarvitaan harvemmin, kuten

1. tekijänoikeusmerkinnät
2. organisaation tiedot
3. tietoa sovelluksesta.

3.1.2 Sivujen navigointialueet

Navigointi tapahtuu joko vaihtoehtoisesti tai yhdistämällä kolmea navigointialuetta. Päävalikko voidaan sijoittaa joko ylätunnisteen alapuolelle tai vasempaan laitaan. Navigoinnissa korostetaan valittua kohtaa jota käytetään. Kuvassa 7 esitetään sivun asemoinnin tavanmukaista rakennetta.



KUVA 7. Sivun asemointi

3.1.3 Työskentelyalue

Työskentelyalue sisältää varsinaisen työalueen, jossa voi olla tekstiä, listauksia, lomakkeita sekä toiminnallisia painikkeita. Työskentelyaluetta suunniteltaessa otetaan huomioon sovelluskohtainen asemointiruudukko, jonka mukaan eri elementit asetellaan eri näytöillä. Yhden sovelluksen sisällä pyritään yhdenmukaiseen toimivaan asetteluun ja sisällön mukaan muutamaan toimivaan asemointimalliin. (Käyttöliittymäsuunnittelun tyyliopas. 2012.)

3.2 Käytettävyyden testaus

Käytettävyyden testauksella tavoitellaan järjestelmässä ilmenevien ongelmien löytämistä. Käyttäjä laitetaan käyttämään ohjelmaa ja pyritään tarkkailemaan käyttäjän toimia ja ohjelman virheitä. Testitilanteessa käyttäjälle annetaan tehtäviä, joista tulisi suoriutua järjestäjien antamien ohjeiden mukaisesti ajatellen ääneen testausta suorittaessa. (Ovaska – Räihä 2012, 13–14.)

Testaustapaukseen liittyy monia ongelmakohtia, kun käyttäjä ihmettelee tehtäviä tehdessään näytöllä näkyviä asioita. Testeissä tulevat helposti esille pienetkin ongelmat. Testattava ei välttämättä huomaa kaikkia ongelmakohtia, vaikka ne voivat olla hyvinkin merkityksellisiä. Testattava voi yrittää suoriutua tehtävistä pitemmän suorituksen kautta, vaikka siihen olisi olemassa suora ratkaisuvaihtoehto. Testattavan ajatellessa ääneen ongelmakohdat löytyvät helpommin. (Ovaska – Räihä 2012, 13–14.)

Testitapauksen suunnittelussa on otettava huomioon, että testattava henkilö on tietotaidoltaan järjestelmän suunniteltua käyttäjäkuntaa. Testitapausten pitää olla samanlaisia kuin oikeassa ohjelmassa olevat tapaukset. Ennen testausta testattavalle on selitettävä testin tarkoitus ja annettava ymmärrys ääneen ajattelusta. Testin aikana testattavalle annetaan tasaisin väliajoin eri tehtäviä tilanteen mukaan ja testattavaa ei saa auttaa ongelmatilanteissa, ennen kuin on selvää, että testattava ei kykene suoriutumaan loppuun tehtävissään. Lopuksi haastatellaan testattavaa ja jutellaan ongelmakohdista. (Ovaska – Räihä 2012, 13–14.)

Videoimalla testitilanne helpotetaan järjestäjän työtä analysoitaessa testiä. Videolta saadaan helpommin selville pienimmätkin yksityiskohdat, kun testitilanteessa ei ole mahdollista tehdä muistiinpanoja. Järjestelmän suunnittelijat eivät yleensä suorita testitilannetta, jonka vuoksi videomateriaali on hyvää materiaalia vakuuttaa suunnittelijat ohjelman virhetilanteista. (Ovaska – Räihä 2012, 13–14.)

Testausta on mahdollista suorittaa hyvinkin aikaisessa vaiheessa ohjelmankehitystä. Paperiprototypointi on hyvä esimerkki tällaisesta testauksesta. Siinä askarrellaan paperi- ja pahvipaloista hyvin alkeellisia, mutta toimivia testiympäristöjä, joista saadaan käyttäjiltä konkreettista palautetta järjestelmän toimivuudesta. (Ovaska – Räihä 2012, 13–14.)

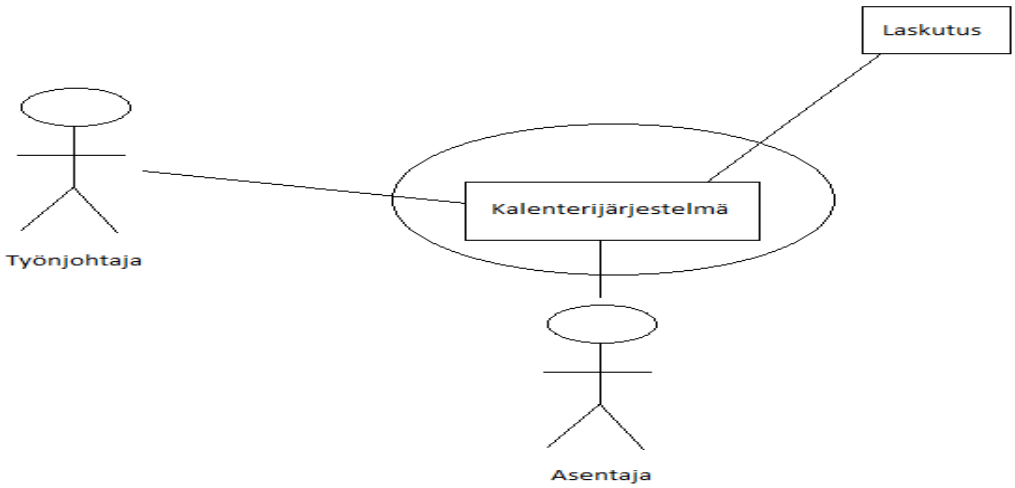
4 TYÖNJOHTAJAN KALENTERIJÄRJESTELMÄ

4.1 Vaatimusmäärittelyn toteutustapa

Vaatimusmäärittelyssä käytettiin Lauesenin (2002) esittämää kaksivaiheista menetelmää: aluksi keskityttiin domain-tason vaatimuksiin ja myöhemmin kriittiseksi koettuihin käyttöliittymän design-tason vaatimuksiin.

Vaatimusmäärittely käynnistyi työnjohtajan haastattelulla. Haastattelun aikana ei ollut tarkoitus löytää heti vaatimuksia, vaan saada mielikuva tilaajan tarpeista ja liiketoiminnallisista tavoitteista. Etukäteen olin valmistautunut kysymään noin kymmenen kysymystä, joihin halusin saada vastauksen haastattelun aikana. Haastattelun aikana tuli selväksi, mikä yrityksen toiminnassa oli aiemmin ollut ongelmallista ja mitä uuden järjestelmän tulisi parantaa. Näitä analysoimme läpi jokaisen sidosryhmän kannalta.

Haastattelun pohjalta päätettiin rajata tuotteen tärkeimpään, välittömään tarpeeseen eli kalenterijärjestelmään. Kun asia oli selvä, aloitettiin elisitoimaan vaatimuksia. Lisähaastatteluilla saatujen tietojen pohjalta lähdettiin analysoimaan järjestelmää ja muodostettiin kontekstikaavio (kuva 8) järjestelmän rajapinnasta. Siitä nähdään, että esimerkiksi laskutus ei kuulu kehitettävään järjestelmään. Kalenterijärjestelmästä laadittiin kontekstikaavio, joka kuvaa järjestelmän sisäisiä ja ulkoisia liityntöjä. Kontekstikaavion avulla voidaan saada järjestelmän todelliset tarpeet esille ja huomataan helpommin järjestelmän laajuus.



KUVA 8. Kalenterijärjestelmän kontekstikaavio

Järjestelmää kuvattiin tämän jälkeen goal-, domain- ja design-tasolla.

Goal-taso

Goal-tasolla kalenterijärjestelmällä tähdätään työnjohtajan työn tukemiseen. Liiketoiminnallinen edellytys järjestelmälle on, että se helpottaa työnjohtajan arkista työtä. Työnjohtajalla on parempi käsitys tulevista projekteista ja asentajien työnohjausta voidaan ennakkoon suunnitella jo useammalle viikolle. Työnjohtaja voi seurata suunnitelmaa kalenterista, joka on saatavilla web-päätelaitteella.

Domain-taso

Domain-tasolla analysoitiin, mitä käyttäjän tehtäviä järjestelmän tulee tukea. Sen perusteella kerättiin käyttäjätehtäviä, joita tuetaan:

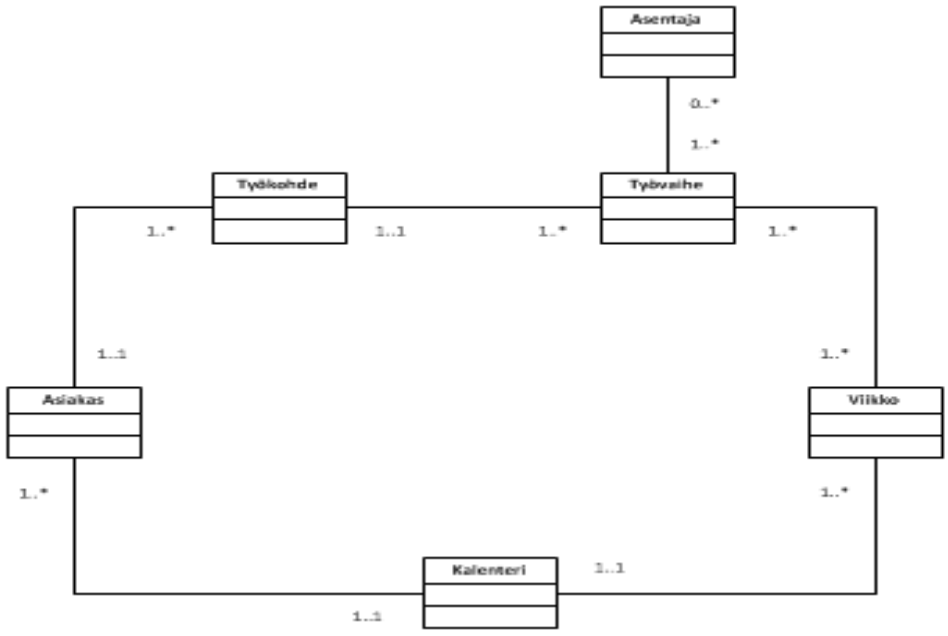
- Käyttäjä kykenee seuraamaan ja tarkastamaan asiakaskohteita.
- Työnjohtaja voi seurata asentajille annettuja tehtäviä viikkotasolla.
- Asentajien tehtäviä voi siirtää suoraan kalenterissa.
- Käyttäjä näkee kaikki tehtävät, jotka on merkattu kyseiselle viikolle.
- Asentaja näkee tehtävät, mutta ei voi muuttaa niitä.

Taulukossa 2 esitetään esimerkki task descriptionista, joka kertoo yhdestä järjestelmään kuuluvasta käyttäjän tehtävän tekstuaalisesta kuvailusta.

TAULUKKO 2. Esimerkki task descriptionista

Tehtävä	**1.5 Määritellään vaiheiden kesto**
Päämäärä	Lisätään päivämäärä vaiheen arvioituun kestoon
Käynnistäjä	Tiedetään kohteen arvioitu kesto
Toistuvuus	Määritellään aina, kun työvaiheita lisätään. Päivittäistä toimintaa.
Kriittisyys	Vaihe ylittää arvioidun määrän
Alitehtävät	
1. Lisätään päivämäärät työvaiheille	
2. Muutetaan päivämäärää	
3. Poistetaan päivämäärät	
Poikkeus	–

Järjestelmän tietovaatimuksia mallinnettiin laatimalla luokkakaavio kuvan 9 mukaan. Luokkakaaviolla kuvataan käyttäjän käsittelemät tiedot ja niiden väliset suhteet.



KUVA 9. Kalenterijärjestelmän luokkakaavio

Tehtävien attribuutit listattiin jokaisen luokan tarpeiden mukaan. Näistä muodostui sitten omat tietovaatimukset.

Kalenterijärjestelmän tukemia käyttäjätehtäviä kuvattiin hyvin täsmällisesti ja ne luetteloitiin järjestykseen. Jokaisesta käyttäjätehtävästä analysoitiin, mikä saa käynnistämään tehtävän ja kuinka usein tehtävää suoritetaan. Jos tehtävään liittyi jotain kriittistä tai puutteita, nämä otettiin huomioon.

Kalenterijärjestelmälle suoritettiin laatutekijöiden kriittisyysanalyysi (taulukko 1). Analyysin perusteella päätettiin, minkä laatuvaatimusten tarkempi määrittely on erityisen tärkeää. Tarkempaa määrittelyä vaativiksi todettiin luotettavuus, käytettävyys ja yhteentoimivuus.

Luotettavuusvaatimuksia ovat mm.

- järjestelmään kirjautuminen
- tietojen tallentuminen järjestelmän kaatuessa.

Käytettävyysvaatimuksia ovat mm. seuraavat:

- Käyttäjän pitäisi oppia käyttämään järjestelmää 15 minuutissa.
- Uuden asiakkaan ja työkohteen luomisessa saa mennä enintään 5 minuuttia.

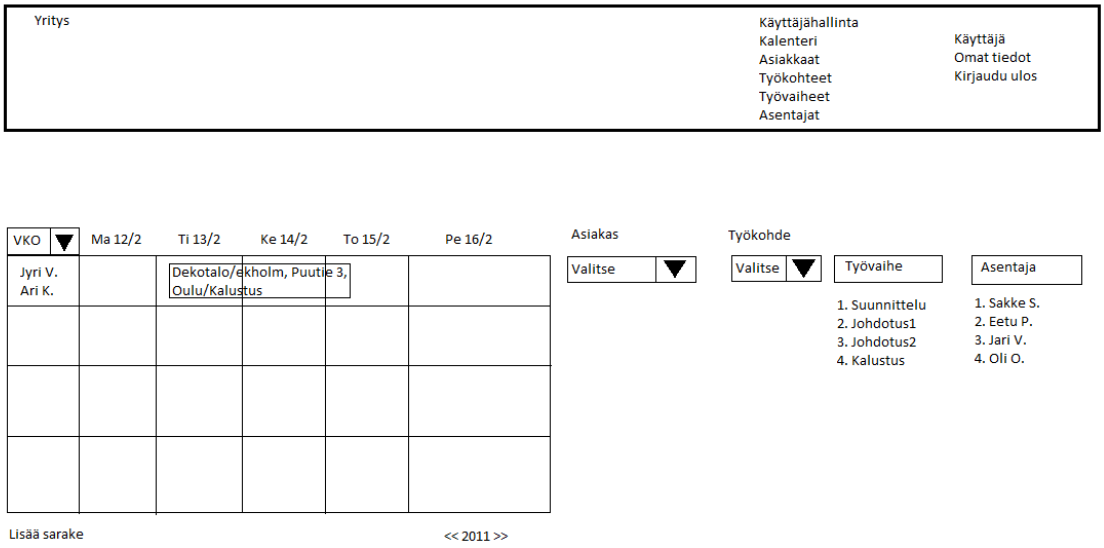
Yhteentoimivuusvaatimuksia on mm. seuraava ominaisuus:

- Järjestelmän rajapinnat on suunniteltava siten, että liitynnät laskutusyms. järjestelmiin voidaan toteuttaa kohtuullisella työmäärällä.

Design-taso

Design-tasolla analysoitiin käyttöliittymän yksityiskohtia. Tuotteessa on oltava kalenteri, jota voi seurata viikkotasolla. Kalenterijärjestelmän pääsivu sisältää kalenterin ja listattuna asiakkaat, työkohteet, työvaiheet, asentajat ja viikot.

Seuraavaksi alettiin suunnitella käyttöliittymän kuvauksia saatujen vaatimusten pohjalta. Järjestelmästä luotiin hyvin yksityiskohtaiset kuvat jokaista vaadittua käyttäjän tehtävää mukaillen. Esittelen kalenterijärjestelmän pääsivun, joka kuvaa, kuten kuvassa 10 näkyy, kalenterin toimintamallin.



KUVA 10. Kalenterijärjestelmän pääsivu

Kalenterijärjestelmän pääsivu, Kalenteri, näyttää välittömästi meneillään olevan viikon ja siihen merkittyjen asentajien tehtävät. Kalenterin vetovalikosta tulevalla viikkovalikolla voidaan seurata, mitä viikoille on merkitty. Asiakkaiden projektit lisätään kalenteriin valitsemalla Asiakas-vetovalikosta asiakas ja Työkohde-valikosta asiakkaan haluttu työkohde. Tämä työkohteen valinta generoi suoraan työvaiheet omaan valikkoon, josta niihin voi tarrautua kiinni ja vetää kalenteriin halutulle asentajaparille ja viikolle. Jokaisesta kalenteriin lisätystä työkohteesta pitää näkyä asiakkaan tärkeimmät tiedot eli Työkohde, Työkohteen osoite ja Työvaihe. Esim. Dekotalo/Ekholm, Puutie 3, Oulu, Kalustus.

Asentajalistauksessa näkyvät kaikki vapaana olevat asentajat, joita ei ole valittu vielä kyseiselle viikolle. Kun valitaan seuraava viikko, pitää asentajalistauksessa näkyä kaikki asentajat, ellei työvaihe ylitä yhtä viikkoa. Asentajan voi lisätä viikolle tarttumalla kiinni ja vetämällä kalenteriin. Kalenterin oikeassa alalaidassa olevaa vuosilukua voi kelata eteen- ja taaksepäin 30 vuotta molempiin suuntiin. Asentajien näkymässä ei näy muuta kuin kalenteri ja siihen merkityt tehtävät.

4.2 Käyttöliittymän prototypointi ja testaus

Käyttöliittymän toimivuutta lähdettiin testaamaan protoilemalla käyttöliittymää paperipaloista rakennetuilla malleilla. Käyttöliittymästä testattiin erityisesti kalenterisovellusta. Tässä tehtävässä asiakas toimi työnjohtajana, jonka tuli suunni-

tella viikon ohjelma kalenteriin saaduilla tiedoilla. Kalenteriliittymänä oli paperiarkki, johon oli piirretty kalenteri. Asiakkaan tuli lisätä paperipaloilla asentajat, työkohteet ja työvaiheet, jolloin ei enää jäänyt yhtään asentajaa toimettomaksi ja kaikki työt saatiin jaettua. Tämä oli erittäin hyvä testi, jossa asiakas huomasi miten, järjestelmä voisi toimia. Loput testattavuustoiminnot liittyivät uuden asiakkaan lisäämiseen ja työvaiheiden lisäämiseen asiakkaalle.

Koko testausprosessi eteni vaihe vaiheelta eteenpäin. Aluksi piirtelin ensimmäisiä kuvia järjestelmästä, joita esittelin asiakkaalle. Asiakas esitti näistä muutostoiveita ja löysi myös selviä virheitä. Näin jatkettiin noin kymmenen kierrosta, kunnes saavutettiin riittävän kypsä lopputulos.

5 POHDINTA

Opinnäytetyön tarkoituksena oli toteuttaa vaatimusmäärittely työnjohtajan kalenterijärjestelmästä Sähkö-Neron järjestelmähankkeeseen. Työn tuloksena syntyi valmis dokumentoitu vaatimusmäärittely yrityksen tarpeista, joilla sitten olisi mahdollisuus lähteä hakemaan joko valmista tai räätälöityä ratkaisua järjestelmästä. Projekti oli haastava, mutta opettava kokemus ja sai hyvän näköalan ohjelmistojen vaatimusmäärittelystä.

Työn aikana opin paljon uutta ohjelmistojen suunnittelusta ja määrittelystä. Nyt ymmärrän paremmin, mitä ohjelmiston määrittely vaatii ennen sen todellista toteutusta. Olen ollut aiemmin toteuttamassa toiminnanohjausjärjestelmiä. Työnjohtajan kalenterijärjestelmä on hyvin samankaltainen, pienempi kuin tavallinen toiminnanohjausjärjestelmä, mutta samoja periaatteita toteuttava järjestelmä. Tämä työ kertoi paljon, miksi aiemmissa töissä olleet toteutukset eivät onnistuneet parhaalla mahdollisella tavalla.

Työn tekeminen oli mielenkiintoista, koska siinä oli kaikki uutta ja järjestelmälle oli asiakkaalla oikeata tarvetta. Järjestelmää määriteltäessä tehtiin paljon yhteistyötä asiakkaan kanssa ja tuli selvitettyä kunnolla, mitä asiakas oikeasti haluaa. Tämä teki työstä paljon hauskempaa.

Projektia tehdessä sai välillä miettiä ihan tosissaan, mitä asiakas oikeasti haluaa ja miten nämä ongelmat voi ratkaista järkevällä tavalla. Asiat loksahtivat pikkuhiljaa kohdilleen ja lopputuloksena oli onnistunut vaatimusmäärittelydokumentti.

LÄHTEET

Carroll J.M. 1995. Scenario-Based Design: Envisioning Work and Technology in System Development. New York: Wiley.

Hull, E – Jackson, K – Dick, J. 2002. Requirements Engineering. New York: Springer, Berlin Heidelberg.

Käyttöliittymäsuunnittelun tyyliopas. 2012. Valtioneuvoston kanslian raportteja 1/2005. Saatavissa: http://vnk.fi/julkaisukansio/2005/r01-kayttoliittymasuunnittelun-tyyliopas/pdf/132202.pdf. Hakupäivä 18.4.2012.

Lauesen, Soren 2002. Software Requirements - Styles and Techniques. Great Britain: Addison-Wesley.

Ohjelmiston vaatimusmäärittely. 2012. Saatavissa: http://fi.wikipedia.org/wiki/Ohjelmiston_vaatimusmaarittely. Hakupäivä 1.4.2012.

Ovaska, S – Räihä, KJ. 1998. Käytettävyystestaus. Systeemityö 4/98, Sytyke ry. Saatavissa: http://www.pcuf.fi/sytyke/lehti/kirj/st19984/13.pdf. Hakupäivä 15.2.2012

Pohl K. 1994. ”The Three Dimensions of Requirements Engineering: A Framework and its Applications”. vol. 19, nro. 3.

Sommerville, I – Sawyer, P. 1997. Requirements Engineering. Chichester: Wiley.

Thayer R.H. – Dorfman M. 1997. Software Requirements Engineering. New Jersey: IEEE Press, Piscataway.

Vuori, M – Kivistö-Rahnasto, J – Toivonen, S 1998. Hyvä käyttöliittymä lähtee käytön tarpeista. Automaatioväylä, nro 7. Saatavissa: http://www.vtt.fi/inf/julkaisut/muut/1998/av7-98.pdf Hakupäivä 15.2.2012

Weinberg G. 1988. Rethinking Systems Analysis and Design. New York: Dorset House.

Wiegers K.E. 1999. Software Requirements. Redmond: Microsoft Press